I mashup e le biblioteche: le tecnologie dietro le quinte

Bonaria Biancu

Introduzione

Da almeno tre anni i *mashup* sono diventati un argomento dibattuto e una tecnologia largamente adottata nel panorama informativo: in quanto applicazioni *situazionali* (devo questa efficace definizione a ReadWriteWeb[1]) proliferano in un ambito, come è ormai il Web 2.0 (l'etichetta è di comodo), ricco di contenuti, relazioni, scambi, collaborazioni. Il remix che il mashup si propone di attuare non è solo da intendere come semplice giustapposizione di contenuti ma come vera e propria (ri)combinazione di dati, anche all'interno di contesti differenti: ne sono un esempio quelle applicazioni che fondono in un unico flusso, informazioni provenienti dal Web e da programmi desktop come i client e-mail o le intranet aziendali.

Le biblioteche, storicamente piuttosto caute di fronte a miscele potenzialmente caotiche di informazioni provenienti da fonti diverse, si sono lanciate con convinzione nell'esplorazione di questa nuova tendenza. È sempre istruttivo consultare a proposito il social network *Mashed Library*,[2] che dà conto di tutte le iniziative su que-

[1] http://www.readwriteweb.com/archives/proto_desktop_mashups.php.
[2] http://mashedlibrary.ning.com/.

JLIS.it. Vol. 1, n. 1 (Giugno/June 2010), p. 133–155.
DOI: 10.4403/jlis.it-26

sto tema che si susseguono in Gran Bretagna dal 2008. Inoltre, la recente tendenza di alcune grandi biblioteche, ad "aprire" i propri cataloghi, rendendo disponibili per il riutilizzo i dati bibliografici (il tesoro nascosto delle biblioteche!), testimonia dell'interesse a creare nuove modalità di utilizzo delle informazioni e della tendenza a una loro sempre maggiore integrazione, a un loro sempre più creativo arricchimento.

Di seguito potete leggere la versione italiana del capitolo "Library mashups: Behind the scenes", pubblicato nel volume *Library Mashups: Exploring New Ways to Deliver Library Data*, edito da Information Today nel 2009.[3] Il capitolo in inglese è disponibile in accesso aperto anche sul *repository* dell'Università di Milano-Bicocca[4]. Invito a leggere il libro non solo per tuffarsi nel mare di informazioni ed esempi pratici che propone, ma anche, perché no, per sostenere un editore che acconsente alla ripubblicazione sul Web dei contenuti dei suoi libri, abbracciando una sfida, quella dell'Open Access, che rappresenta un altro grande e positivo fermento del mondo dell'informazione. La riproposizione dei contenuti del capitolo è praticamente integrale, tranne per quel che riguarda l'ultimo paragrafo, quello dedicato agli *editor* per mashup. Popfly, l'editor di Microsoft recensito nella versione inglese del capitolo, è stato dismesso per confluire nel più ampio progetto del Microsoft Web Platform, un ambiente per lo sviluppo e l'*hosting* di applicazioni web. Stessa sorte è toccata a Google Mashup Editor, a favore di Google App Engine, un ambiente simile al concorrente.

Yahoo! con il suo Pipes, il primo dei *mashup editor* a irrompere nella scena del Web e a registrare un successo che non declina ma anzi aumenta con il passare del tempo, sembra essere rimasto l'unico grande player interessato allo sviluppo di una piattaforma dedica-

[3] http://books.infotoday.com/books/LibraryMashups.shtm. Al libro è dedicato anche un blog, a cura dell'editor Nicole Engard, http://mashups.web2learning.net/.

[4] http://hdl.handle.net/10281/5117.

ta eminentemente all'utente medio di Internet per la creazione di mashup. A *Pipes* dedica un intero capitolo la curatrice del volume, Nicole Engard, pertanto vi rimando alla lettura del suo contributo per un approfondimento delle funzionalità di questo strumento.

Concludo esprimendo la mia soddisfazione per la pubblicazione di un articolo dedicato ai mashup all'interno del primo numero di una rivista di scienze dell'informazione che si preannuncia seria e rigorosa ma anche aperta e curiosa del nuovo. A Mauro Guerrini, che mi ha voluta testimone di questo battesimo, va la mia affettuosa gratitudine.

Le API: il segreto dei mashup

Secondo la definizione di *ProgrammableWeb*, un mashup comporta la combinazione di dati provenienti da due o più fonti web al fine di creare nuovi servizi e applicazioni.[5] Per creare un mashup è quindi essenziale poter ricavare informazioni strutturate dai siti web (o almeno poter convertire in informazione strutturata il contenuto del sito una volta estratto). Il meccanismo più efficace per accedere a un servizio web è probabilmente la sua *Application Programming Interface* (API). Le API sono un insieme di funzioni e procedure per accedere a un servizio web: esse mostrano la logica sulla quale il servizio è costruito, le sue risorse chiave e le funzioni passibili di essere invocate da programmi esterni al sito. In altre parole, le API consentono a un programma di accedere a un *web service* e di manipolare i dati esposti, nello stesso modo in cui l'interfaccia web di un sito permette agli utenti di navigare al suo interno ed esplorarne il contenuto.

[5]La stesura di questo capitolo è debitrice a Karen Coyle e Raymond Yee per il loro generoso aiuto e supporto.

Ogni sito decide come esporre il suo contenuto verso l'esterno, perciò possono esservi grandi differenze tra fornitori di API – così come per lo stesso provider in relazione ai suoi differenti servizi. Naturalmente, documentare pubblicamente le API e acconsentire a un loro libero e gratuito utilizzo, rende più semplice ai programmatori sfruttarle per i propri scopi. Ma perché un sito dovrebbe fornire API pubbliche ed esporre le proprie informazioni – quindi la fonte del proprio valore – gratuitamente online? Innanzitutto, perché permettere agli utenti di manipolare intensivamente i contenuti di un sito web o di invocare i suoi servizi da un *client* di un'applicazione esterna, è un ottimo modo per testare l'applicazione: quando centinaia (o addirittura migliaia) di utenti cominciano a sviluppare servizi web sfruttando i contenuti di un sito, questo è di fatto sottoposto a un test su larga scala, e presumibilmente i bachi contenuti nell'applicazione verranno scoperti e sanati.

Per ottenere informazioni sulle API è consigliabile consultare la sezione "Sviluppo" (o "*Developer*", in inglese) del sito, all'interno della mappa del sito, oppure le pagine di Help o ancora quelle delle FAQ; per converso, i siti che mettono a disposizione interfacce pubbliche di accesso per gli utenti, dovrebbero assicurarsi che esse siano visibili sui portali che raccolgono informazioni sulle API (vedi più avanti in questo capitolo). Un aspetto molto importante da tenere in considerazione quando si adopera un'API è la licenza con cui è rilasciata o i cosiddetti "*Terms of Service*" (TOS), che rappresentano le condizioni stabilite dal provider del servizio. Possono infatti esservi limitazioni sul tipo di utilizzo che gli sviluppatori possono fare di un'API: nella maggior parte dei casi, le API pubbliche sono gratuite per finalità non commerciali, mentre per utilizzi di tipo commerciale il fornitore potrebbe richiedere il pagamento di un costo oppure bloccare del tutto l'accesso al servizio. Inoltre, esistono spesso limitazioni in ordine all'utilizzo di banda e al numero delle richieste

che vengono indirizzate verso il servizio. È anche utile notare che le API evolvono nel tempo e potrebbero cambiare: è pertanto necessario verificare di tanto in tanto la documentazione fornita dal provider. Un altro problema cui prestare attenzione è il copyright sui contenuti utilizzati nel mashup: quanto viene pubblicato online potrebbe non essere legalmente riutilizzabile da altri oppure agli sviluppatori potrebbe essere richiesto di implementare l'applicazione secondo specifici requisiti, per ottenere il permesso di ripubblicare o comunque riutilizzare le informazioni presentate online. Per tutte queste ragioni è necessario esaminare con accuratezza la licenza con la quale le informazioni sono state rilasciate e le condizioni alle quali possono essere utilizzate da terze parti. Laddove manchi una licenza specifica, è da presumere che valgano le consuete disposizioni di legge del Paese entro cui si sta operando. In aggiunta alla logica applicativa che permette agli sviluppatori di conoscere quali risorse siano rese disponibili e quali operazioni possano essere effettuate su esse, una caratteristica delle API sono i protocolli di comunicazione adoperati: come, da un punto di vista tenico, è possibile invocare l'interfaccia di programmazione perché lo scambio di informazioni abbia luogo? Quali sono i requisiti previsti per la costruzione delle richieste verso le API? Ma anche: in quali formati l'informazione verrà restituita al client che ha eseguito l'interrogazione? Quali canali possono o devono essere usati perché lo scambio di contenuti sia rapido e sicuro? Per meglio comprendere e per cercare una risposta a queste questioni, è arrivato il momento di dare un'occhiata al mondo dei web service e alle sue molteplici funzionalità.

Architettura dei Web Service

Il W3C definisce un *web service* come «a software system designed to support interoperable machine-to-machine interaction over

a network».[6] I *web service* sono dunque una tecnologia che abilita lo scambio di informazioni e di comunicazioni tra differenti applicazioni. Comprendere come i web service funzionino è chiaramente un requisito fondamentale per il corretto uso delle API e la creazione di mashup efficaci.

I web service sono basati su un modello concettuale che prevede un *service provider*, cioè un'applicazione che pubblica le informazioni e/o che fornisce la possibilità di eseguire determinate operazioni, e un *service consumer*, cioè un client che fa uso delle informazioni e/o dei servizi resi disponibili. Il *service consumer* (per esempio il sito nel quale il mashup avviene) esegue una richiesta verso un *service provider* (ovvero per esempio il sito che fornisce le API). Se la richiesta è effettuata nella maniera corretta, il *service provider* invia una risposta formattata secondo le caratteristiche del servizio.

Nella pratica, il protocollo di trasporto e il linguaggio più usati sono HTTP (HyperText Transfer Protocol) e XML (eXtensible Markup Language). Se si lavora a un programma che deve acquisire informazioni, dati o servizi da un sito web attraverso un'API, è necessario conoscere i particolari requisiti tecnici di quel sito e costruire la richiesta di conseguenza. Una volta formattata secondo i criteri richiesti dal sito web, la richiesta viene inviata attraverso uno dei metodi del protocollo HTTP: GET (quello che viene usato *implicitamente* nella normale navigazione in Internet), POST (adoperato primariamente dal service consumer per *inviare* informazioni verso il *service provider*), PUT o DELETE (per particolari operazioni che devono essere effettuate sul server). Gli *stili* utilizzati nell'architettura dei *web service* possono essere diversi in relazione alle tecnologie e ai protocolli impiegati: i più diffusi sono REpresentational State Transfer (REST) e SOAP (sigla che originariamente rappresentava l'acronimo di 'Simple Object Access Protocol', ma che

[6] http://www.w3.org/TR/ws-arch/#whatis.

ormai ha acquisito significato autonomo), che rappresentano le basi, rispettivamente, delle architetture *resource-oriented* e *service-oriented*.

REST è il più semplice, e quindi il più utilizzato, protocollo nella creazione di mashup. Come vedremo nel paragrafo "Un esempio con Yahoo! Answers" (p. 147), REST consiste di interfacce basate su un'architettura resource-oriented, molto semplici da implementare. Concettualmente REST prevede un servizio che richiede informazioni o esegue delle operazioni da/su una specifica applicazione. La richiesta utilizza un URL contenente i parametri dell'API, ed è trasmessa usando uno dei metodi HTTP (per esempio GET). Da notare che l'architettura REST richiede che le operazioni siano invocate propriamente attraverso i metodi HTTP, e non all'interno dell'URL inviato in Rete, come invece spesso accade, dal momento che l'URL dovrebbe contenere soltanto la cosiddetta *scoping information* (ovvero le *risorse* richieste e non le *operazioni* da eseguire). Questo significa che, in uno stile *pure REST*, se si desidera semplicemente richiedere informazioni da un sito, occorre usare il metodo HTTP GET; se invece si richiede al *service provider* di eseguire delle operazioni sulle sue risorse, allora la richiesta dovrebbe servirsi del metodo apposito – tipicamente POST, PUT, o DELETE.[7] Sebbene la maggior parte dei *web service* dichiarino di essere RESTful, essi spesso espongono i propri dati in modalità ibride (per esempio inseriscono i metodi dell'API all'interno dell'URL invece di utilizzare correttamente il protocollo HTTP).[8]

Quando una richiesta viene inviata attraverso la Rete, il *service provider* fornisce una risposta contenente informazione formattata

[7] Le differenze tra lo stile REST *puro* e stili come il REST-RPC sono ben evidenziate nel libro *RESTful web services* (Richardson e Ruby 2007)

[8] Cfr. a questo proposito anche i post di Duncan Cragg: *STREST (Service-Trampled REST) will break Web 2.0*, http://duncan-cragg.org/blog/post/strest-service-trampled-rest-will-break-web-20/ e *The REST dialogues* http://duncan-cragg.org/blog/tag/dialogue/

in un linguaggio di rappresentazione (spesso in XML, sebbene altri formati, come JavaScript Object Notation (JSON), siano sempre più utilizzati). Il client può quindi usare la risposta come input per altre operazioni, oppure presentarla graficamente in una pagina web. Come si può notare, la logica di richiesta-risposta dei *web service* di tipo REST è la stessa che viene usata sul web quando un utente naviga sui vari siti online. Le uniche differenze sono che, da un lato, le transazioni sono attivate attraverso chiamate alle API invece che attraverso URL inviati in un browser, e, dall'altro, che il formato della risposta è pensato per essere *compreso* (visualizzato, manipolato etc.) da un programma, invece che da un essere umano.

SOAP è un altro stile che nel mondo dei *web service* si è sviluppato accanto a REST; esso poggia su standard e protocolli internazionali ed è stato adottato soprattutto nel mondo enterprise. SOAP usa HTTP come protocollo di trasporto per lo scambio di informazioni ma richiede che, sia la richiesta inviata dal service consumer sia la risposta fornita dal *service provider*, siano "impacchettate" in un contenitore XML. Lo stesso provider descrive il *web service* attraverso specifici XML schema, che vengono poi pubblicati online, in modo che le applicazioni che li utilizzeranno, possano conformarsi ad essi.

La maggior parte dei siti 2.0 usano interfacce REST per le loro API. Ciò è comprensibile se si pensa che uno degli obiettivi del Web 2.0 è abbassare quelle barriere tecniche che in passato hanno impedito all'utente medio di partecipare attivamente alla produzione di informazione online. È piuttosto semplice costruire una richiesta attraverso l'impostazione dei parametri in un'URL e il suo invio verso un server, da un browser o un form web; ma è ancora più facile se l'URL prende GET come metodo: anche con scarse conoscenze di programmazione, oggi creare un mashup potente e funzionale è possibile grazie allo stile REST.

SOAP è senz'altro più complesso da implementare e, secondo

quanto sostenuto dai fan di REST (i cosiddetti *RESTafarians*), fallisce nel promuovere la facilità d'uso e l'efficacia implicite nel concetto di World Wide Web. È però importante comprendere come funziona l'architettura SOAP, sia perché molti siti usano solo SOAP, sia perché SOAP è più standardizzato ed elaborato di REST. Peraltro non è infrequente che tra i requisiti di un progetto di sviluppo *business-oriented* vi sia proprio la capacità di lavorare con SOAP.

Dove trovare API online

Quando ci si avvicina al mondo dei mashup per la prima volta, è molto utile poter avere accesso alle esperienze di altri utenti, così come conoscere i siti aperti al riutilizzo delle informazioni. Una risorsa di grande utilità da considerare per questo tipo di *training* è *ProgrammableWeb*.[9] Questo sito presenta attualmente il più ampio database di mashup; esso raccoglie anche news dal web, tendenze dal mondo dell'industria, informazioni tecniche, guide e riferimenti vari. Ogni mashup in *ProgrammableWeb* ha associata una descrizione delle funzionalità che lo caratterizzano, insieme con l'elenco delle API usate, *tag* che ne descrivono i contenuti, informazioni sull'autore e link a mashup collegati. Inoltre, gli utenti di *ProgrammableWeb* possono votare i mashup proposti, aggiungerli ai propri preferiti e seguire attraverso i feed RSS altre applicazioni che fanno uso delle stesse API.

Come ci spiega Raymond Yee nel suo eccellente libro *Pro Web 2.0 mashup*[10] (Yee 2008b), una maniera utile per sviluppare idee creative che potranno essere propedeutiche alla realizzazione di un mashup, è studiare le applicazioni già create da altri utenti e provare a rispondere a domande come le seguenti:

[9] http://www.programmableweb.com
[10] http://blog.mashupguide.net

- Cosa viene combinato in questo mashup?
- Perché proprio questi elementi vengono combinati?
- Dove avviene il remix o la ricombinazione delle informazioni?
- Come i vari elementi vengono combinati (e cioè, primariamente all'interno dell'interfaccia del mashup ma anche dietro le quinte di un'applicazione)?
- Come è possibile estendere il mashup? (Yee 2008a)

Queste domande e le loro risposte forniscono un'utile griglia attraverso la quale analizzare i mashup presenti su *ProgrammableWeb* e aiutano ad approfondire le dinamiche e le problematiche coinvolte nella creazione di un mashup.

È possibile imparare qualcosa del mashup anche studiando le informazioni convogliate dai componenti che abilitano il mashup, in particolare le API – le vere protagoniste dei mashup. Su *ProgrammableWeb* a ogni API è associato uno schema analitico che fornisce una descrizione generale, i *tag*, le ultime novità, un elenco di mashup che ne fanno uso, i protocolli implementati (*web service* di tipo REST e SOAP, con i relativi formati di output dei dati), le funzionalità (per esempio i diversi metodi da invocare), i protocolli di sicurezza adottati (autenticazione, SSL etc.), il supporto (offerto sia dal provider sia dalla comunità degli utenti), e infine i termini di utilizzo. Spesso sono presenti anche guide su come usare l'API e i feedback di chi l'ha già testata.

Le informazioni fornite da *ProgrammableWeb* rappresentano dunque una insostituibile risorsa nel fornire una rappresentazione dell'ampio panorama di siti che acconsentono all'utilizzo dei propri servizi e contenuti da parte di terzi. In più, i principianti apprezzeranno la semplicità delle informazioni offerte dal sito e la possibilità

di effettuare ricerche nel database anche per librerie di codice e tool di sviluppo.

ProgrammableWeb è una directory generalista e, tra le diverse migliaia di mashup censiti, sono poche le applicazioni *library-oriented*. Grazie all'impegno del movimento della Library 2.0 e al rilascio di software di *information management* modulari e compatibili con gli standard più diffusi a livello internazionale, in tempi recenti i mashup e le API si sono conquistati una sempre crescente attenzione nel mondo dei professionisti dell'informazione. Il *Library Software Manifesto*[11] richiama l'importanza di API pubbliche per le biblioteche che acquistano software di tipo ILS (Integrated Library System). È infatti solo grazie all'esistenza di interfacce pubbliche di accesso che tutti i dati – bibliografici e non – racchiusi nei database delle biblioteche possono essere utilizzati al di là degli usi previsti dai fornitori (naturalmente parliamo pur sempre di usi legali!). Sebbene le biblioteche, in quanto organizzazioni che lavorano primariamente (con) le informazioni, possano sembrare di diritto i protagonisti principali dell'universo di dati, mashup e *web service*, in passato non si è registrata grande consapevolezza a riguardo, né intorno all'importanza di API aperte né relativamente al potenziale che esse hanno per le biblioteche e i loro progetti o obiettivi. Per esempio, non esistono molte API per i cataloghi online e, al di là delle *query* previste dal protocollo Z39.50, è piuttosto difficile, se non impossibile, creare un mashup come il SOPAC[12] di John Blyberg, senza avere un'API che lo supporti. Spesso i bibliotecari non sanno se il software in uso nella propria biblioteca dispone di API e i *vendor* non sempre forniscono spontaneamente questo tipo di informazioni.

Questo scenario rende gli sforzi di social network come *Mashed Library*[13] ancora più importanti e meritevoli di attenzione. Il gruppo

[11] http://techessence.info/manifesto
[12] http://www.thesocialopac.net
[13] http://mashedlibrary.ning.com

che ha dato vita a questa rete, ha raccolto un elenco di API e di *web service library-oriented*, poi incorporato in forma permanente all'interno del blog TechEssence.[14]

Tra le API elencate su TechEssence, diverse riguardano gli OPAC, così come i software per la gestione delle risorse digitali – e sono tutte rese disponibili da alcuni tra i maggiori player del mondo dell'Information Technology, come OCLC, il sito di e-commerce Amazon, il fornitore di software per biblioteche Talis e il servizio di *social bookmarking* dedicato ai libri LibraryThing.[15] È da ricordare a questo proposito anche il *JISC Information Environment Service Registry*,[16] un catalogo di risorse elettroniche e repository digitali, che documenta *web service* e API con i relativi requisiti, fornendo al contempo modalità di accesso e interrogazione attraverso i protocolli standard OAI-PMH, SRU, OpenURL e Z39.50.

È insomma giunto il momento di stimolare e supportare le iniziative bottom-up delle biblioteche e del mondo LIS in generale, e di sfruttare (al) meglio la creatività dei bibliotecari per fare un uso degli applicativi più flessibile e al passo coi tempi. È tempo di sfruttare intensivamente gli Integrated Library System e indirizzarli verso un maggiore orientamento all'utente, condizionando positivamente le modalità con le quali le diverse fonti di contenuti digitali espongono i loro dati per rendere l'informazione disponibile ad essere usata da applicazioni di terze parti come i blog, i calendari online o i feed RSS. In questo modo gli stessi utenti saranno messi nelle condizioni di creare applicazioni che né i vendor né i bibliotecari avrebbero mai immaginato.

Se, dopo aver letto questo capitolo, deciderete di "sporcarvi le mani" con lo sviluppo dei mashup, potete considerare sia Program-

[14] http://techessence.info/apis
[15] http://www.librarything.com
[16] http://iesr.ac.uk

mableWeb sia TechEssence come un buon punto di partenza per la scoperta della magia delle API.

Mashup senza API

Abbiamo descritto fin qui le API e i loro protocolli di comunicazione, ma che accade se un sito non dispone di interfacce strutturate attraverso le quali servizi esterni possano recuperare le informazioni di cui hanno bisogno? Sebbene esistano diversi siti che offrono interfacce di programmazione, non tutti le rendono pubblicamente disponibili per il riutilizzo di terze parti; in alcuni casi è possibile utilizzarle solo a pagamento, mentre in altri le aziende tengono le API segrete poichè esse vengono considerate strategiche per il successo commerciale di un prodotto o di un servizio. Parimenti, esistono siti web che non offrono nessun tipo di interfaccia con cui interagire. In questi casi è più difficile utilizzare i dati, ciò nonostante possono esservi comunque buone *chance* di creare un mashup.

Raymond Yee nel suo libro invita a studiare innanzitutto le specifiche modalità di costruzione degli URL (*URL language*) usate dal sito di interesse. Questo è utile quando anche un'API è disponibile ma diviene cruciale nei casi in cui non esistono API da utilizzare ed è necessario scoprire autonomamente l'architettura delle informazioni adottata dal sito. Il modo in cui gli indirizzi web sono costruiti può aiutare a capire se e come l'informazione è strutturata in risorse identificabili da URL, oppure categorizzata, taggata o comunque organizzata in modo tale che un programma sia in grado di operare con essa.

Una fonte affidabile di dati formattati in maniera appropriata e che consente il loro ri-utilizzo all'interno di *web service* è il *feed* di un sito web. I *feed* non sono altro che piccoli pezzi di informazione rappresentati in XML o in uno dei suoi dialetti (come RSS o ATOM),

normalmente usati dai siti per disseminare gli aggiornamenti dei propri contenuti e letti dai navigatori per il mezzo di aggregatori o *feed reader*. In effetti i *feed* possono essere visti come un semplice *web service* RESTful, dal momento che una risposta formattata in XML è invocata da una richiesta formulata attraverso un URL trasmesso su HTTP. Utilizzando il file che risiede sotto il tipico pulsante arancione che indica la presenza di un *feed*, lo sviluppatore può analizzare le informazioni e usarle come input per un'altra applicazione o *renderizzarle* direttamente in una pagina web. La presenza di funzionalità di import/export di contenuti o l'utilizo di tecnologie di sviluppo web come AJAX (linguaggio basato su JavaScript e XML usato per la creazione di applicazioni web dinamiche), sono altri indizi che il sito dispone di dati processabili da un programma esterno. Se un sito non fornisce nè un chiaro *URL language* nè *feed* o altre modalità di accedere ai suoi dati, allora occorre affidarsi semplicemente alla sua interfaccia web, usando la tecnica nota come screen scraping. Attraverso questo meccanismo, le informazioni concepite esclusivamente per essere visualizzate e usate da esseri umani (gli utenti del sito), vengono estratte dalle pagine web e inviate come input a un programma. L'applicazione di tipo *screen scraper*, quindi, agisce sul livello più superficiale delle pagine web, ovvero il livello di presentazione dei contenuti, e usa i *tag* HTML come agganci per identificare le risorse informative desiderate.

Naturalmente, queste tecniche di *hacking* producono risultati spesso instabili e inaffidabili: se un sito non mette a disposizione un'API, possono esservi solide ragioni - incluse quelle legali - perchè esso non renda disponibili i propri dati per l'utilizzo da parte di applicazioni esterne. È quindi necessario analizzare accuratamente se lo *screen scraping* può essere usato in una particolare situazione, anche se esso rappresenta l'unica opportunità di ottenere informazioni da riutilizzare.

Un esempio con Yahoo! Answers

Passiamo adesso dalla teoria alla pratica e vediamo come un vero *web service* è in grado di effettuare richieste da una fonte online e di processare i dati che ottiene in risposta per renderli utilizzabili all'interno di un mashup. Dal momento che un mashup combina dati provenienti da due o più fonti, per ciascuna di esse occorrerà comprendere le peculiari modalità di input e output dei contenuti e delle informazioni, come vedremo in questo paragrafo con Yahoo! Answers. Esistono molti siti che rendono le proprie API disponibili per gli utenti: motori di ricerca come Google, che offre l'opportunità di interrogare direttamente il suo indice; siti di media sharing come Flickr[17] e YouTube,[18] dai quali è possibile ottenere filmati e foto; aggregatori come Technorati[19] e Feedreader,[20] che foniscono contenuti *user-generated* e *feed*, e molti altri. Yahoo! Stesso offre una grande varietà di API e *web service* che interagiscono con il suo motore di ricerca con altri servizi online come le mappe, la musica, l'informazione finanziaria, il social bookmarking e così via. Nello Yahoo! Developer Network (YDN)[21] è possibile trovare tutte le informazioni necessarie sulle interfacce delle applicazioni disponibili e su altre tecnologie collegate, insieme con guide e video-tutorial, librerie per il web design, una galleria di mashup, report sulle iniziative di *hacking* sponsorizzate e infine il supporto sia dell'azienda sia della community degli sviluppatori. All'interno di questa molteplicità di servizi, abbiamo scelto di occuparci in questo capitolo di Yahoo! Answers.[22]

[17] http://www.flickr.com
[18] http://www.youtube.com
[19] http://www.technorati.com
[20] http://www.feedreader.com
[21] http://developer.yahoo.com
[22] http://answers.yahoo.com

Yahoo! Answers è una «online community where anyone can ask and answer questions on any topic. Yahoo! Answers connects people to the information they're seeking with those who know it».[23] Consultando la sezione YDN di Answers[24] scopriamo che sono disponibili API per l'accesso e il riutilizzo di domande e risposte pubblicate dagli utenti su Yahoo! Answers. Le API usano un'interfaccia di tipo REST: per quanto detto sopra, se desideriamo interrogare questa interfaccia abbiamo bisogno di:

1. costruire un'URL secondo i parametri specificati dall'API,
2. inviarlo utilizzando uno dei metodi HTTP richiesti.

Se si vogliono usare le API di Yahoo!, occorre richiedere innanzitutto un Application ID.[25] Questa procedura è spesso richiesta dai siti che mettono a disposizione le proprie API, perchè in questo modo il *service provider* può tracciare il numero e il tipo di richieste e la banda di cui ogni applicazione fa uso, in aggiunta alle caratteristiche del client che richiede i contenuti.

Ci sono quattro tipi di *query* disponibili per l'interazione: *questionSearch, getByCategory, getQuestion e getByUser*. Le *query* restituiscono, rispettivamnte: le domande che trattano dell'argomento scelto nella *query*; le domande archiviate sotto una certa categoria; dettagli di tutte le risposte fornite a una domanda; le domande postate su Yahoo! Answers da uno specifico utente. La pagina dedicata alle API ha anche un form web attraverso il quale è possibile testare le *query* e verificare le risposte fornite dal servizio. L'unico metodo HTTP disponibile per questa API è GET; dunque è possibile solo leggere le informazioni dal sito, non modificarle, caricarne di nuove o eliminare dati sul server.

---

[23] http://help.yahoo.com/l/us/yahoo/answers/overview/overview-55778.html

[24] http://developer.yahoo.com/answers

[25] http://developer.yahoo.com/wsregapp

La prima *query*, *questionSearch*, "finds open, resolved, or up-for-vote questions that include your search terms". Per accedere alle informazioni desiderate, è necessario costruire una richiesta che recuperi le domande pubblicate su Answers, utilizzando i parametri forniti dall'API. Il target URL a cui la richiesta va indirizzata è:

http://answers.yahooapis.com/AnswersService/V1/questionSearch

dove *http://answers.yahooapis.com/* è l'*hostname* dell'API e *AnswersService* il nome del servizio. L'acronimo *V1*, posizionato di seguito, si riferisce al numero di versione dell'API e *questionSearch* rappresenta il tipo di *query* che abbiamo scelto. Alcuni tra gli argomenti che possono essere utilizzati per questo tipo di richieste sono:

- *query*: ricerca i termini (obbligatorio)
- *category_id*: ricerca solo in specifiche categorie attraverso gli ID corrispondenti (gli ID possono essere individuati nell'URL navigando tra le categorie di Yahoo! Answers)
- *region*: filtro basato sulla nazione (per esempio, us: United States; uk: United Kingdom;it: Italy e così via)
- *sort*: imposta l'ordinamento nel set dei risultati per
  - relevance: rilevanza (default)
  - date_desc: data discendente (le domande più recenti per prime)
  - date_asc: data ascendente (le domande più vecchie per prime) (si può omettere per l'opzione di default: relevance)
- *appid*: l'*application ID* (obbligatorio)

- *output*: definisce l'output della call. Valori accettati sono - xml, json, php e rss (si può omettere per l'opzione di default: "xml")[26]

Per esempio, per ottenere tutte le domande chieste da utenti statunitensi riguardanti l'energia solare (solar energy) all'interno della sotto-categoria *Green Living*, e ottenere i risultati ordinati per data con i più recenti in cima, occorrerebbe costruire il seguente URL:

> *GET http://answers.yahooapis.com/AnswersService/V1/questionSear ch?appid=MyYahooAppId&query=solar%20energy&category_id=21 15500307®ion=us&sort=date_desc*[27]

(da notare che la stringa MyYahooAppId va sostituita con il proprio identificativo, e che gli argomenti e i loro valori vanno *URL-encoded*).[28] Il numero 2115500307 in *category_id* è l'identificativo della categoria *Environment/Green Living* category, come mostrato nell'URL della corrispondente pagina web di Yahoo! Answers. Di default il massimo numero di risultati è dieci e il formato di presentazione dei risultati è l'XML. Dal momento che non abbiamo specificato nessuno di questi due parametri, il servizio di Yahoo! utilizzerà le impostazioni pre-impostate. È possibile testare questo URL

---

[26]Gli argomenti e le loro spiegazioni sono riportati dalla pagina *questionSearch* (http://developer.yahoo.com/answers/V1/questionSearch.html)

[27]Vedi Yahoo! Developer Network per ottenere informazioni su come costruire URL di tipo REST (http://developer.yahoo.com/search/rest.html)

[28]La *URL encoding* (o *Percent encoding*) è una tecnica usata per convertire i caratteri speciali presenti in un URL, in un formato valido. Per esempio, lo spazio tra le parole in composti come energia solare (solar energy), viene codificato nell'URL come %20. Per maggiori informazioni si veda la pagina dedicata all'argomento su Wikipedia (http://en.wikipedia.org/wiki/Percent-encoding) e un elenco di caratteri codificati (www.w3schools.com/TAGS/ref_urlencode.asp)

incollandolo nella barra degli indirizzi del browser:[29] il risultato sarà una risposta formattata in XML contenente le informazioni richieste più un insieme di valori ulteriori come l'*id* e il *nick name* dell'utente che ha posto la domanda, il numero di risposte fornite per ogni domanda e - se esiste - la risposta scelta come migliore. Il codice XML può essere analizzato e inviato come input a una applicazione di terze parti oppure trasformato in una pagina HTML per mostrare i risultati nel proprio stile grafico preferito. Per esempio, un consorzio di biblioteche su scala nazionale o regionale, potrebbe cooperare nei servizi di reference sviluppando un mashup che raccolga all'interno di un'unica interfaccia domande su un certo argomento, provenienti sia dagli utenti del servizio di reference online delle biblioteche sia dagli utenti di Yahoo! Answers.

È anche possibile ottenere un feed in risposta scegliendo RSS come formato di output (*&output=rss*). In questo caso l'URL che risulta dalla chiamata alla API è l'indirizzo di un feed RSS che può essere usato da un feedreader per ottenere notifiche ogni qual volta vengono pubblicate su Answers nuove domande che rispondono ai requisiti pre-impostati.

Il formato JSON restituisce un JavaScript serializzato, un formato più lineare dell'XML che è fornito di default. JSON può essere usato in combinazione con la funzione di *callback* resa disponibile tra i parametri dell'API per risolvere problematiche di sicurezza come la Same Origin Policy, che si presentano quando le applicazioni per il mashup utilizzano linguaggi di programmazione client-side come

[29]È possibile invocare un'API da un'applicazione scritta in un linguaggio server-side come Perl e PHP orppure da un client Ajax. Oppure si possono provare le funzionalità dell'API inviando la richiesta attraverso un programma a linea di comando come cURL o attraverso l'estensione per Firefox Poster. Yahoo! Developer Network offre un Software Development Kit (SDK) (http://developer.yahoo.com/download/) con librerie di codice disponibili per utilizzo pubblico, per implementare *web service* in diversi linguaggi di programmazione

JavaScript. Bisogna poi tener presente che questo *web service* limita le *query* per indirizzo IP a 5.000 al giorno e che richiede obbligatoriamente la sottoscrizione degli *Yahoo! API Terms of Use*[30] e *Terms of Service*.[31] Infine, i siti web e le applicazioni che usano questo servizio di Yahoo! devono mostrare in chiaro la dicitura "*Powered by Yahoo! Answers*".

Editor per Mashup

Un vantaggio significativo dell'attuale livello di attenzione intorno ai mashup, è il fatto che ci sono sempre più *tool* e servizi che rendono la creazione di un mashup un processo veramente semplice, anche per l'utente medio di Internet. Alcuni di questi eliminano interamente il ricorso alla programmazione, nascondendo i dettagli tecnici e presentando interfacce semplificate. Alcune grandi compagnie hanno cominciato a lavorare agli editor per mashup, ovvero ad applicazioni che permettono di ricombinare le informazioni attraverso semplici comandi all'interno di un'interfaccia grafica. Come spiega Nicole Engard nel capitolo del libro "Library Mashups: Exploring New Ways to Deliver Library Data", Yahoo! con il suo Pipes[32] ha rivoluzionato il panorama dei mashup editor: con la semplice giustapposizione di box contenenti gli indirizzi Web dei servizi che si intende utilizzare, Pipes consente di combinare contenuti testuali e multimediali provenienti da fonti diverse, dando luogo a un feed RSS.

Tra gli altri editor con interfaccia grafica, vanno ricordati Mashup Maker[33] e Dapper[34], così come tool più *business-oriented* come JackBe

[30] http://info.yahoo.com/legal/us/yahoo/api/api-2140.html
[31] http://info.yahoo.com/legal/us/yahoo/utos/utos-173.html
[32] http://pipes.yahoo.com
[33] http://mashmaker.intel.com/web/index.php
[34] http://www.dapper.net

Presto Platform[35] e Serena[36].

In genere, però, a coloro i quali muovono i primi passi nel mondo del mashup e hanno limitate competenze informatiche, è consigliato cominciare con un editor "visuale" e in questo caso Yahoo! Pipes, con la sua copiosa documentazione, gli esempi, i numerosi moduli già creati a disposizione e, *last but not least*, un'ampia community a corredo, è senz'altro il candidato ideale.

Riferimenti bibliografici

ARKIN, ASSAF (2006), «Scraping with style: scrAPI toolkit for Ruby», http://blog.labnotes.org/2006/07/11/scraping-with-style-scrapi-toolkit-for-ruby/, http://blog.labnotes.org/2006/07/11/scraping-with-style-scrapi-toolkit-for-ruby/.

BLOCH, JOSHUA (2006), «How to design a good API and why it matters», http://www.slideshare.net/guestbe92f4/how-to-design-a-good-a-p-i-and-why-it-matters-g-o-o-g-l-e, http://www.slideshare.net/guestbe92f4/how-to-design-a-good-a-p-i-and-why-it-matters-g-o-o-g-l-e.

CAMPBELL, RYAN (2006), «How to add an API to your Web service», http://particletree.com/features/how-to-add-an-api-to-your-web-service/, http://particletree.com/features/how-to-add-an-api-to-your-web-service/.

FOX, PAMELA (2006), «Web 2.0 mashups: how people can tap into the 'Grid' for fun and profit», http://www.slideshare.net/wuzziwug/web-20-mashups-how-people-can-tap-into-the-grid-for-fun-profit-20924/, http://www.slideshare.net/wuzziwug/web-20-mashups-how-people-can-tap-into-the-grid-for-fun-profit-20924/.

GARRETT, JESSE J. (2005), *Ajax: A New Approach to Web Applications*, http://www.adaptivepath.com/ideas/essays/archives/000385.php, http://www.adaptivepath.com/ideas/essays/archives/000385.php.

GREGORIO, JOE (2004), *How to Create a REST Protocol*, http://www.xml.com/pub/a/2004/12/01/restful-web.html, http://www.xml.com/pub/a/2004/12/01/restful-web.html.

[35] http://www.jackbe.com/products

[36] http://www.serena.com/mashups

He, Hao (2004), *Implementing REST Web Services: Best Practices and Guidelines*, http://www.xml.com/pub/a/2004/08/11/rest.html, http://www.xml.com/pub/a/2004/08/11/rest.html.

Heilmann, Christian (2004), *Beginning JavaScript with DOM scripting and Ajax: from novice to professional*, Berkeley: Apress.

Herren, John (2007), «Introduction to mashup development», http://www.slideshare.net/jhherren/mashup-university-4-intro-to-mashups, http://www.slideshare.net/jhherren/mashup-university-4-intro-to-mashups.

Levitt, Jason (2005), *JSON and the dynamic script tag: easy, XML-less Web services for JavaScript*, http://www.xml.com/pub/a/2005/12/21/json-dynamic-script-tag.html, http://www.xml.com/pub/a/2005/12/21/json-dynamic-script-tag.html.

MacManus, Richard (2007), *Proto Enables Desktop Mashups - Also Giving Away iPods in Mashup Contest*, http://www.readwriteweb.com/archives/proto_desktop_mashups.php, http://www.readwriteweb.com/archives/proto_desktop_mashups.php.

Richardson, Leonard e Sam Ruby (2007), *RESTful web services*, Sebastopol: OReilly.

Schnell, Eric (2007), «Mashups and Web services», in *Library 2.0 and beyond: innovative technologies and tomorrow's user*, a cura di Nancy Courtney, Westport, Conn.: Libraries Unlimited, pp. 63–74.

OASIS (2005), *SOA reference model*, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm.

Theurer, Dan (2005), *Web services + JSON = dump your proxy*, http://www.theurer.cc/blog/2005/12/15/web-services-json-dump-your-proxy/, http://www.theurer.cc/blog/2005/12/15/web-services-json-dump-your-proxy/.

Udell, Jon (2004), *The beauty of REST*, http://www.xml.com/pub/a/2004/03/17/udell.html, http://www.xml.com/pub/a/2004/03/17/udell.html.

Yee, Raymond (2008a), *Pro Web 2.0 mashups : remixing data and web services*, Berkeley: Apress.

— (2008b), *Semantic Search the US Library of Congress*, http://blog.programmableweb.com/2008/04/29/semantic-search-the-us-library-of-congress/, http://blog.programmableweb.com/2008/04/29/semantic-search-the-us-library-of-congress/.

Zakas, Nicholas C. (2004), *Beginning JavaScript with DOM scripting and Ajax: from novice to professional*, Berkeley: Apress.